# 《C/C++学习指南》

## 第14.2讲：引用的更多用法

作者：邵发　　QQ群：417024631
官网：http://afanihao.cn

## 引用的更多用法

(1)引用作为函数的参数
(2)引用作为函数的返回值

# 引用作为函数的参数

和指针类似，引用也可以作为函数的参数，功能相同。

```
void test (int& a)
{
   a = 999;
}
int main()
{
   int number = 0;
   test(number);
   return 0;
}
```

“传引用”和“传地址”本质相同：
相当于对参变量作为了一个初始化 int& a = n;

# 引用作为函数的参数

“传引用”和“传地址”本质相同：

所以，参数的传递有两种方式：
(1)传值 （效率低）
(2)传地址或传引用（效率高）

注：回顾第10章对二者效率的比较

# 引用作为函数的返回值

和指针一样，引用也可以作为函数返回值。【不易理解，较晦涩】
比如，

```
int number = 0;  // 全局变量
int& test()
{
	int& a = number;
	return a; // 返回引用
}
int main()
{
	int& r = test();
	r = 123;  // 修改目标对象的值
	return 0;
}
```

# 引用作为函数的返回值

（1）演化

```
int number = 0;  // 全局变量
int& test()
{
	return number; // 此处作了简化
}
```

return number：并不是返回了number的值，而返回了它的引用。

# 引用作为函数的返回值

（2）返回值作为左值

```
int main()
{
	test() = 123;
	return 0;
}
```

注：普通函数的返回值都只是右值，只有返回引用时才能当做左值来用（不过可读性不高）

# 引用作为函数的返回值

（2）返回值作为左值,再来一个例子

```
struct Object
{
	int id;
	char name[16];
};
Object one; // 全局变量
Object& test()
{
	return one; // 返回全局变量的引用
}
int main()
{
	test().id = 10; // 函数的返回值是引用，可以作为左值使用
	return 0;
}
```

## 安全问题

和指针一样，引用也有安全性问题。

主要是检查：
引用的目标对象是否有效？

比如，以下引用的目标对象是一个局部变量，那么，在函数退出后，目标对象失效，所以引用也就不能再用了。

```
int& test()
{
    int target = 123;
    return target;
}
```

## 小节

引用的两个主要用法：
（1）作为参数
（2）作为返回值（较难理解）